Immersive Authoring of Tangible Augmented Reality Applications

Gun A. Lee [α] Claudia Nelles [β] Mark Billinghurst [β] Gerard Jounghyun Kim [α]
[α] *Virtual Reality Laboratory, Pohang University of Science and Technology*
[β] *Human Interface Technology Laboratory New Zealand, University of Canterbury*
[α] *{endovert, gkim}@postech.ac.kr* [β] *{claudia.nelles, mark.billinghurst}@hitlabnz.org*

Abstract

In this paper we suggest a new approach for authoring tangible augmented reality applications, called 'immersive authoring.' The approach allows the user to carry out the authoring tasks within the AR application being built, so that the development and testing of the application can be done concurrently throughout the development process. We describe the functionalities and the interaction design for the proposed authoring system that are specifically targeted for intuitive specification of scenes and various object behaviors. Several cases of applications developed using the authoring system are presented. A small pilot user study was conducted to compare the proposed method to a non-immersive approach, and the results have shown that the users generally found it easier and faster to carry out authoring tasks in the immersive environment.

1. Introduction

In recent years augmented reality (AR) has emerged as an important medium for education and entertainment. As the number of people building AR applications grows, it becomes increasingly apparent that a need exists for more efficient development tools. Most current AR applications are built using low level programming with dedicated tracking and graphics libraries, rather than from concrete "components" that content developers or artists are more used to. In this paper we describe a high level toolkit that enables rapid development of AR applications with no programming.

Our toolkit is based on the Tangible AR input metaphor. Tangible AR [8] is an approach that combines tangible user interface [9] input methods with AR display and output. In this way the virtual content in the AR interfaces can be manipulated using physical objects, making these interfaces extremely intuitive. For example, in the VOMAR interface [7] a real paddle is used to pick up and place virtual furniture in a simple scene assembly program. Even complete novices were able to use the VOMAR application with ease.

Tangible AR interfaces rely on accurate tracking of real objects. To achieve this we use the ARToolKit [1] computer vision tracking library which can calculate the position and orientation of a camera relative to square fiducial markers. ARToolKit makes the development of Tangible AR applications easier, yet it is still a software library that requires programming skills to be used. In contrast, desktop interactive multimedia contents can be easily built using various authoring tools, such as the Microsoft PowerPoint, or the Macromedia Director and Flash. These tools do not require the user to be an experienced programmer.

Programs such as PowerPoint are relatively easy to use because they incorporate direct manipulation input with WYSIWYG (What You See Is What You Get) output. First used for desktop publishing, WYSIWYG editing is incorporated into most 2D authoring systems, providing fast, concurrent evaluation of the layout.

The aim of our research is to develop an AR authoring interface that is as easy to use as desktop WYSIWYG interfaces. Our authoring tool uses Tangible AR interaction techniques and an authoring method called 'immersive authoring'. Immersive authoring is an authoring method that allows the direct specification and testing of the content within the execution environment. Thus the development environment and the execution environments are the same, and the authoring environment provides the full experience of the building contents by itself.

In the rest of the paper, we review previous work on authoring tangible augmented reality applications and interfaces that supported immersive authoring within immersive virtual reality environments. As a first step in developing the authoring environment, we analyzed the target application domain to define a Tangible AR application model to be used later in our authoring system. Based on this application model, we designed interaction methods for the authoring tasks and developed a prototype immersive authoring system. To illustrate the usability of this type of interface, we

present several examples of Tangible AR applications built with our authoring system. In addition we compare our approach to a non-immersive authoring system in terms of the ease of use and task performance.

2. Related work

Creating an interactive virtual world consists of two major tasks. One is modeling the geometry (or form) of the virtual objects, and the other is describing the virtual object behaviors and interactions.

Since it is a straightforward approach to use 3D interactions for modeling 3D geometries, there have been various attempts to construct (or model) a virtual world geometry within immersive virtual environments [4][13][15][20].

In contrast, there have been few attempts to define the behaviors while immersed in the virtual environment. Stiles and Pontecorvo [19] suggested a conceptual virtual reality system called 'Lingua Graphica' that uses visual languages to program the virtual environment itself. However, the system remained as a concept and was never implemented. Steed and Slater [18] developed a system that visualized the links of data flow between the virtual objects, making users able to manipulate them directly. Lee et al. [11] pointed out that behavior authoring tasks could also benefit from direct 3D interactions and proposed to use a programming by demonstration approach for authoring virtual object behaviors.

There have also been some attempts to construct virtual worlds from within AR environments. Poupyrev et al. [17] suggested a rapid prototyping tool for modeling virtual aircraft cockpits. The system provided a set of virtual gauges and instruments that can be copied over physical tiles. The users were able to test the layout using an AR interface. Kato et al. [7] suggested a generic interaction method for manipulating virtual objects within AR environments. They also showed their interaction method working in an AR modeling system for arranging furniture in a virtual room. Piekarski and Thomas [16] suggested 3D geometry modeling techniques for outdoor AR systems. The modeling system was for constructing virtual representations of physical landmarks while investigating the outdoor scenery. However, all of this work focused on modeling forms (or geometry) of virtual worlds within AR environments, and did not address authoring behaviors and interactions.

In contrast with previous researches that have focused on 2D desktop authoring and then testing in AR environment [5][6][14], our research is focused on developing an AR tool that enables users to author AR contents (especially, the behaviors and interactions) from 'within' the AR interface. One particularly valuable approach for such a tool is to base it on the Tangible AR design principles. We describe this further in the next section.

3. Application domain analysis

Tangible AR interfaces [8] are those in which 1) each virtual object is registered to a physical object and 2) the user interacts with virtual objects by manipulating the corresponding physical object. As the definition implies, there are mainly two kinds of entities in Tangible AR applications: virtual objects and physical objects.

Virtual objects are the main entities that the users are interested in and want to interact with. They are visualized in various forms, such as 2D images or text and, of course, 3D geometries. On the other hand, physical objects serve as tangible interaction points on which the visualization of virtual objects are overlaid, and where the user inputs are sensed. Tangible AR applications typically use real physical objects as input devices.

Since the interaction between virtual objects and the user is mediated by physical objects, there should be logical connections between the physical and virtual objects; physical objects that the user physically interacts with and virtual objects that the user virtually interacts with. In order to draw a virtual object registered on a physical object, the position and orientation data of physical objects are needed to be fed to the corresponding virtual objects.

However, the connection between physical and virtual objects can be more than a direct mapping between their property values. For example, suppose that we want to change the size of a virtual object according to the proximity of the physical object to the user's view. The distance should be obtained by calculating the norm of the relative position vector between them, and this requires a couple of arithmetic operations. To represent these logical (or arithmetic) operations, we introduce another type of entity named 'logic box.' A logic box might represent a single logical (or arithmetic) operator, or even a complex behavior such as controlling joint angles of a virtual character.

Putting all these features together, we suggest a *component based application model* for Tangible AR applications. In our model, a Tangible AR application is described with a number of components and connections between their properties. Using the data flow model to describe the user interface for virtual environments traces back to an early virtual reality

program named Body Electric [10] from VPL. And it also agrees with the previous work of Steed and Slater [18] that investigated on the immersive authoring in a virtual environment.

There are three types of components in our application model: the physical object, the virtual object and the logic box. Each component has a set of properties that represents the state of the component, and each of these properties differs between different component types. Each property has a specific data type and read/write attribute according to the features of the component it represents. For instance, the position property of a physical object has a vector type value that represents a three-dimensional coordinate. Its value can be read but can't be modified freely, since it is determined by the physical location of the physical object. Table 1 summarizes the properties of each type of components. Properties of physical objects are mainly related to their tracking results. The visibility of the physical object represents whether the object is successfully tracked, and the transformation, position and orientation properties represent the physical pose of it. Virtual objects have similar properties with physical objects, while they have writable attributes, meaning they can be freely modified. Some virtual objects representing sound sources also have boolean properties for playing the sound, in addition. Properties of logic boxes vary from one another. They are determined by the logical functions that the logic box represents. For instance, a vector addition logic box might have two input and one output vector properties, while a logic box representing a motor like behavior might have only a single property that gives the rotation value as the time flows.

Table 1. Properties of each type of component

component type	property name	data type	attribute
physical object	visible	boolean	r
	transformation	matrix	r
	position	vector	r
	orientation	vector	r
virtual object	visible	boolean	r/w
	base transformation	matrix	r/w
	transformation	matrix	r/w
	position	vector	r/w
	orientation	vector	r/w
	scale	scalar	r/w
	play (optional)	boolean	r/w
logic box	-	-	-

AR applications can be developed by connecting the components together. Properties of components can be connected to each other when they have compatible data types and attributes. For example, properties with scalar data types can be linked to those with scalar or boolean values, but cannot be linked to those with vectors, unless they are modified to a scalar value using a logic box. A property used as a target must be writable, while the readable attribute is sufficient for source properties.

Once a property is linked to another, its value is updated according to the source property. For instance, a virtual object can be registered to a physical object simply by connecting the transformation attribute of the virtual object to that of the physical object. Typically, we introduce another property named 'base transformation' to virtual objects to represent the parent reference frame of the object.

4. Immersive authoring design

Given the application model, we now analyze the task requirements for authoring tangible augmented reality applications. After specifying these requirements, we describe our interaction design chosen to fulfill the requirements.

4.1. Task analysis

The authoring task can be regarded as building an application by describing it with the established application model, i.e. defining entities and filling out their property values declared in the model. Since our application model is a component based one, the main authoring task will be manipulating the components. Table 2 summarizes the main tasks and their subtasks for manipulating components.

Table 2. Main tasks of component manipulation

Main task	Subtasks
Create	Select type
Destroy	Select a component to destroy
Modify	Select a component to modify Browse & select a property Change the value of the property
Connect (or Link)	Select components to connect Browse & select properties Connect/disconnect the properties

The most basic tasks are creating and destroying the components. For creating a component, the user needs a way to specify the type of the component s/he wants to create. Users need to browse through a list showing what kind of components they can define.

This requires a menu-like interface to a set of items that users could browse through before choosing one of them. Some vital components, such as pre-defined physical objects, could exist without the need for being explicitly created. These components will be provided to the user from the beginning of the authoring process and the users would be able to use them immediately. Destroying a component requires the ability to select a component. Users need to select a component which they want to destroy, and this requires an interface for pointing to or selecting a specific virtual object.

The created components may need to be modified to complete the AR application. According to the component model described in section 3, modifying a component is simply changing its property values. Prior to changing the component property value, a user needs to select the component and its property that s/he wants to change. This requires an interface for browsing over the list of properties and their values. After the property is selected, users need to specify a new value for it. The interface for specifying a component property value may vary according to the data type of the property. For example, simple scalar values are easy enough to modify with buttons or keypads while 3D transformations may be more conveniently modified with direct manipulation.

The last main task of component manipulation is to connect their properties with each other. Similar to changing the property values, users first need to select components and the properties they want to connect or disconnect. Hence, the same interface could be reused for selecting properties, while a new interaction method is needed for specifying the status of their connection.

4.2. Design guidelines

Prior to designing the interaction methods for our tangible authoring interface, we begin by presenting design guidelines for immersive authoring systems.

First of all, the most important feature of an immersive authoring system is the concept of '*What You Feel Is What You Get (WYFIWYG)*.' This refers to the ability to feel all the sensory elements (visual, aural, and other elements if there are any) of the final content as it is being constructed. The main point of immersive authoring is to be able to experience the virtual worlds while they are being built. Therefore, immersive authoring systems are presumed to provide fast (or even concurrent) evaluation of the resulting content, a.k.a. WYFIWYG.

Taking advantage of *direct 3D manipulation* is the next guideline for designing interactions for immersive authoring. Since the augmented reality environment implies interaction in three-dimensional real space, direct 3D manipulation of virtual objects is easy and efficient within immersive authoring AR environments.

The third guideline is to maintain the *application model transparency*. Although direct manipulations are efficient for three-dimensional object manipulations, they might hide the details of the underlying application model. Users may need to explicitly assign specific values, such as the X coordinate of the object position, as well as to grab and drop it directly in the position and orientation they want to place it. Therefore, the system must provide transparent interfaces (or interaction methods) that show the details of the underlying application model, so that the users will have the ability to directly manipulate them.

Finally, the interaction methods and interfaces for immersive authoring must be as similar as possible to the ones used in the target application domain being authored. We refer to this property as *consistency*. Adding different interfaces to the authoring environment implies context switching of the developers' mental activity, and might distract their attention, delaying the development process. Distracting the user's attentions might not only cause temporal delays to the development process, but also degrade the quality of the authoring virtual world. For instance, the presence, one of the most important quality measures of virtual (or augmented) environments, is degraded by distractions and makes it hard for developers to fully experience the authored virtual world and correctly evaluate it. Therefore, it is highly recommended to use similar (or at least non-conflicting) interfaces with the target application domain.

4.3. Interaction design

In this section we illustrate our suggested interaction designs for each of the subtasks from Table 2. In order to maintain the consistency between the authoring environment and the final application, we avoided introducing new environmental setups. Instead, we only introduced props for the authoring task that can be used in the same environment with general Tangible AR applications. The physical props are simple pads and cubes that are commonly used in Tangible AR applications. Figure 1 shows three basic props used for the authoring task: a component browser, a manipulator and a disposer. Since only these props are added to the tangible AR application being built, the users are guaranteed to concurrently experience the final application without any disturbance throughout the development task, and this meets the 'WYFIWYG' design guideline.

For creating a new virtual object component (or logic box components), users need to select the type of virtual object they want to create. The component browser provides a physical interface for browsing over available 3D virtual objects and selecting the desired one. Users can browse over the models one by one, by pressing (pointing) [12] the arrow buttons on the both sides of the browser. To create a new virtual object, users point at the target 3D model for a second with the cube manipulator (shown in Figure 2).

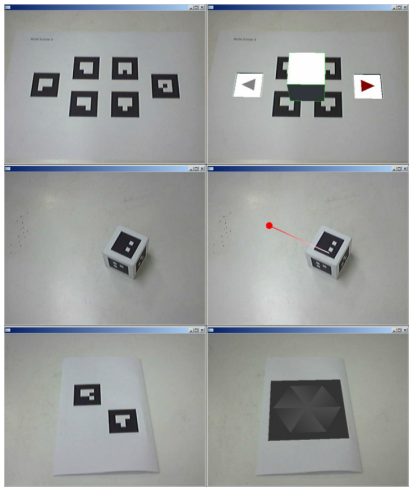

Figure 1. The props for authoring task: component browser, manipulator and disposer

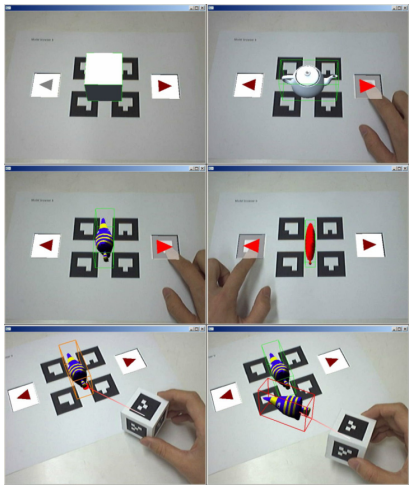

Figure 2. Creating a new virtual object

After a virtual object is selected with the manipulator prop, it moves according to the movement of the manipulator. The selected virtual object is kept in a fixed position relative to the manipulator when it is selected, and rotates according to the pose of the manipulator. To release (or unselect) the virtual object, the user simply needs to hide the manipulator for a couple of seconds. The virtual object will remain in the last position and orientation where it was placed. This interaction was designed following the notion of the 'drag and drop' metaphor, which is one of the most well known direct manipulation methods in 2D desktop graphical user interfaces.

The picking up and dropping interaction method is used for destroying objects, as well as for placing (or modifying) them. The upper row of the Figure 3 shows moving a virtual object from one physical object to another, while the lower row shows destroying it by dropping on the disposer prop.

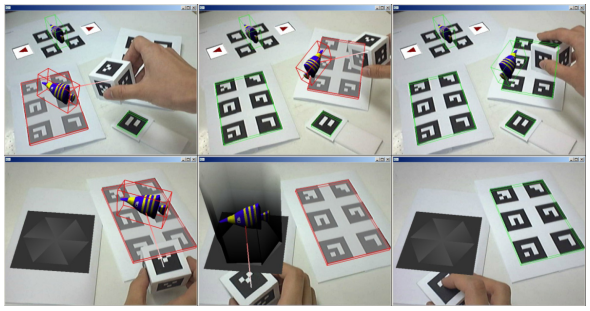

Figure 3. Pick & drop interaction for moving and destroying virtual objects

Although picking up an object and dropping it on a desired position and orientation takes advantage of the direct 3D manipulation, it hides the details of how the underlying application model is affected: the base transformation and visible properties of the moved virtual object are connected to the transformation and the visible properties of the physical object where the object is dropped, and the position and the orientation properties of the virtual object are changed in order to place the virtual object in an appropriate position relative to the physical object. Therefore to provide the model transparency, two more types of interfaces, inspector pads and keypads, were added. These elements provide detailed information about the selected components and let the users tweak them (see Figure 4).

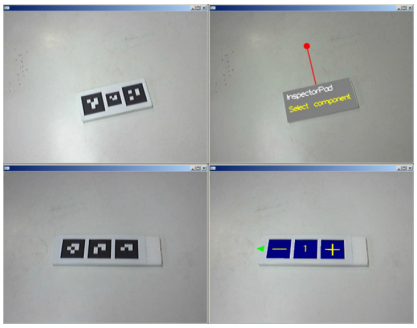

Figure 4. Inspector pad and keypad

The interaction for selecting and deselecting a component with an inspector pad is similar to that of manipulators: pointing at a component for a second with the probe and hiding the interface. While the manipulators are only allowed to select virtual object components, users can also select physical objects with the inspector pads.

Once a component is selected, the inspector pad shows the properties and their values of the selected component (see Figure 5). The users can browse through the properties by holding and manipulating the inspector pad. The list of properties shows up when the inspector pad is close enough to the users' view, and the list can be scrolled up and down by tilting the inspector pad up and down. The property displayed on the middle of the inspector pad is selected when the inspector pad is moved away, and the inspector pad shows the value of the selected property. The display format of the value is changed according to its data type, and the read/write attributes are represented by the green arrows on each side of the inspector pad.

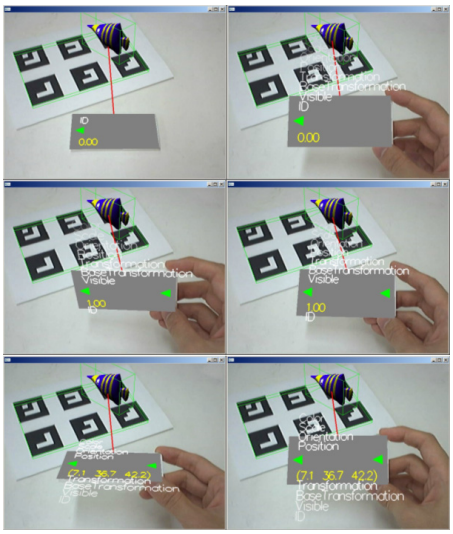

Figure 5. Browsing through the properties and their values of a component with an inspector pad

To change the value of the selected property, users can use a keypad together with the inspector pad. Since most of the properties can be represented by numeric values, keypads are used for providing an input method for these. We designed a keypad using occlusion based interaction [12], the same interaction method applied to the model browser. A number of visual markers used for tracking the prop are also used for the button pressing interaction. Figure 4 shows an instance of the keypads that has '+/-' buttons together with a unit selection button on the middle. Users can select the unit between 0.1, 1 and 10, and by pressing the '+/-' buttons, the value is raised or lowered by the selected unit. To change the value of the property selected on the inspector pad, the user connects the keypad to the inspector pad, and operates the keypad to modify the value. Figure 6 shows an example of using an inspector pad and a keypad to change the scaling property of a cube virtual object component.

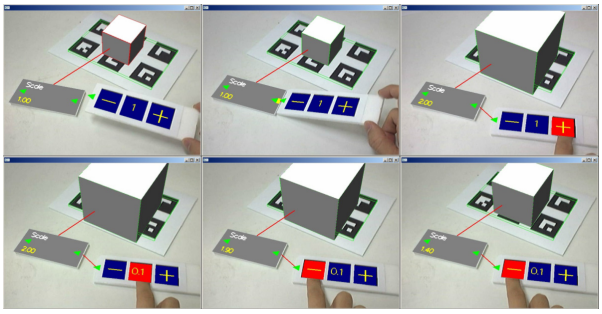

Figure 6. Changing scale property value with an inspector pad and a keypad

Connecting object properties implies selection of multiple properties. Instead of introducing another selection method, here we simply duplicated the inspector pad to select two object properties being connected.

The interaction method for connecting two selected properties can be designed in a various ways. We've first tried to directly map the logical connection between properties to the physical connection between inspector pads with puzzle cut edges. Although the physical connection worked as an easy and intuitive input method, it was not feasible to use it for displaying the current connection status, since they were not controllable in an automatic manner. In addition, direct mapping of physical and logical connections was poor to prevent incompatible connections, e.g. connecting properties with incompatible data types or attributes. To solve these problems, we've altered the interaction design to toggle between connected and disconnected states when two edges of inspector pads were contacted. Each vertical edge of the inspector pads were used as an input and output port of the selected property. And by contacting these edges together, a link was made (or destroyed) between them if the selected properties were compatible. The same method was used for connecting keypads and inspector pads (see Figure 6).

Figure 7 shows an example of connecting properties of two components. The visibility property of a virtual fish is connected to the same property of a physical paddle, making the fish disappear when the paddle is hidden by the user's hand.

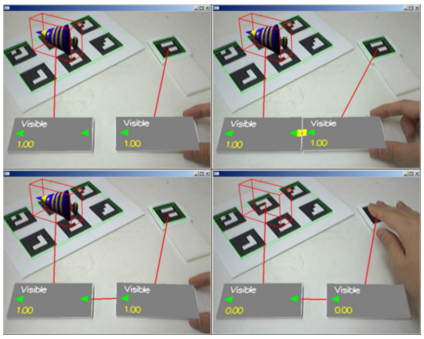

Figure 7. Connecting properties

5. Implementation

The authoring system described in this paper was developed on a consumer level personal computer. The computer was running Windows XP operating system on a Pentium 4 processor with 1GB main memory. A GeForce4 3D graphics card from NVIDIA was used to accelerate the OpenGL graphics processing.

For tracking physical objects, we used a vision based tracking method. The ARToolKit [1] software library was used for calculating the 3D position and orientation of the visual markers, and a plain USB web camera from Logitech was used to acquire video images for the tracking. The capturing resolution was set to 320x240 and the shutter speed was 30 frames per second. The camera was mounted on a head mounted display to provide a real world view to the user, forming a video see-through AR configuration.

We used our custom 3D model loader, based on the OpenGL library, to visualize the 3D graphics contents and the virtual authoring tools. To make the interaction easier for selecting components with the manipulator and inspector pads, bounding boxes are visualized around the component objects, and their colors are changed according to their status: normal, pointed and selected. These bounding boxes are only shown when there are authoring props within the user's view.

6. Case studies and discussion

6.1. Development cases

To show the efficiency and feasibility of using our immersive authoring method, we have constructed several example Tangible AR applications.

The first example is a simple scene with a windmill (see Figure 8). The scene consists of three virtual objects: the ground, a tower and a vane. It took about a minute to place the virtual objects and check that every thing was placed in the right place. A logic box representing a rotation behavior was used to specify the vane to spin around. The logic box was set invisible for viewing. It totally took less than 3 minutes total to construct the whole scene, connect the properties to define the behavior, and to validate the final product.

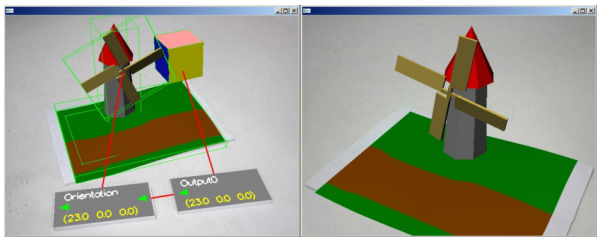

Figure 8. An example application with animation

In addition to passive animations of virtual objects, interactive features can also be added. Figure 9 shows a sequence of images, constructing an interactive Tangible AR application similar to the Shared Space application [2]. The application shows two tiles with a virtual object on each, a hare and a tortoise for example. The user can examine the virtual objects by manipulating the tiles on which they are anchored. When two tiles are brought close together, different models are shown, such as the hare and the tortoise greeting each other (see the last row of Figure 9).

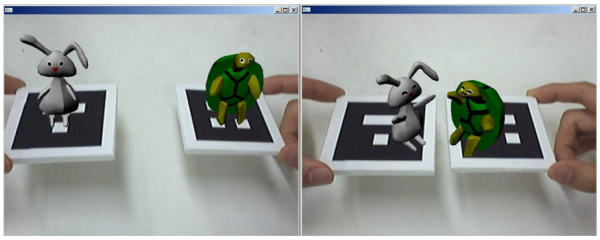

Figure 9. An interactive Tangible AR application

To build this application, four virtual objects were needed: the normal and greeting posed models for the hare and tortoise. First, the virtual objects were placed on two physical tiles, one for the hare and another for the tortoise. The visibilities of the virtual objects were controlled by the proximity value of the physical tiles. In order to check the distance and to control the visibilities, we used a logic box with a special function. The logic box had two input properties of position, and output properties with a boolean value that represented whether the two input positions were close enough or not. By connecting position properties of the two tiles to the logic box input, and connecting 'near' and 'far' boolean output properties of the logic box to four virtual objects' visibility, properly, the application was completed. About 5 minutes were needed for building and testing the whole application.

The last example application is an interactive storytelling book application, similar to the MagicBook [3] (see Figure 10). We used one of the popular stories of Aesop, 'The race between a hare and a tortoise.' The story consists of three main scenes: starting the race, the hare taking a nap and the tortoise winning. To add interactivity to the story line, we made a decision point on the second scene to let the users choose whether the hare should sleep or not. According to the user's decision, the winner on the last scene would be determined differently.

Thirteen pre-modeled 3D objects were brought in and three sheets of paper with printed markers were used as the book pages. To implement the interactive feature, special properties for occlusion based interaction were added to the physical object component: a set of boolean valued properties indicating which button (i.e. marker) was pressed (for the last). These properties were connected to the visibility of the characters placed on the final scene, selecting different endings of the story according to the user's decision. It took about 15 minutes to construct the scenes and to connect object properties for implementing the interaction.

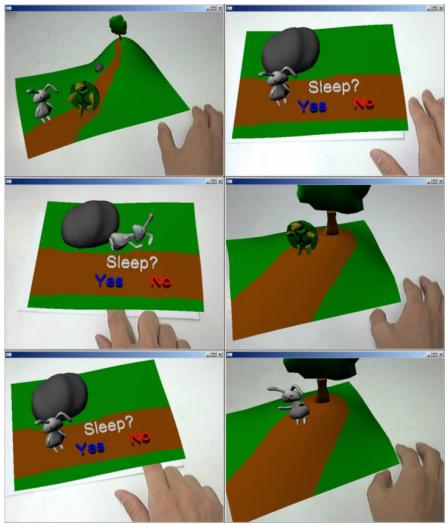

Figure 10. An interactive storytelling application

6.2. User Study

We have conducted a pilot usability test of our authoring interface. The test has been held at the end of a series of workshops in which participants learned 3D modeling. Each of these workshops ran for 3-4 days, and the participants were children (9-14 years old) and their parents. In these workshops, the participants created virtual contents and then used our authoring tool to add these contents to an AR scene. It should be noted that the participants were not experts in programming or 3D modeling, and had not experienced augmented reality before. Despite this, they were all able to create an AR scene by the end of the workshop. The participants responded they couldn't believe that it could be so easy to create AR scenes and they particularly enjoyed using the immersive authoring interface to manipulate their models.

The user study was to compare the usability of our immersive authoring tool with another desktop AR authoring tool, CATOMIR [21]. The main aims of the study were to find out how these authoring tools would be accepted by the users, how efficient each of them would be for the participants to use, and where both of them were showing usability faults and how to get rid of them.

Similar to our immersive authoring tool, CATOMIR is also aimed to allow non-programmers to create AR applications. It is also based on a similar component based model to represent the AR contents. However, it uses more traditional mouse, keyboard and desktop screen-based interfaces. Users can create components by selecting them from a list in 2D graphical user interface. The property values of a component can be investigated and modified with a dialog box styled interface. Users can also link the properties by dragging between the two properties they want to connect, e.g. visibility properties of the tracking marker component and the 3D geometry component. Positioning and rotating the virtual objects can be achieved with a dialog box interfaces where the user types in the numeric values or pushes +/- buttons to change the transformation values. In addition, it also provides a simple assistant tangible prop for translating and rotating task. However, users still need the mouse to set the mode (translation or rotation) and the axis while using the assistant prop. After constructing a compound of components in a 2D desktop authoring environment, users can run and test their application with the AR interface.

There were 24 participants (16 male and 8 female), ranging in age from 9 to 50 years who were novices in 3D graphics and programming. The participants had a training phase before the test, where they trained on each authoring tool until they were confident in using it. During the training phase the participants learnt to create an AR scene by loading 3D models, placing them on a specific marker and bringing it into a specific position. Each participant tried the authoring tools in a different sequence to prevent the study being influenced by their previous experiences. After they were comfortable with using the tools, the task for the main test was given.

Each participant used both tools and was given the same test task with each, although with different contents (3D models). The task was to load a specific model, to put it onto a specific marker and to bring it into a specific position, which they've practiced during the training phase.

The time for the task completion was measured and the number of errors was counted. After performing the tasks the participants were asked to fill in a questionnaire and answer interview questions. The questions were to gather further information about where problems with the programs occurred, which tool they preferred and how they felt about using each tool.

The average speed for the whole task participants who used our immersive authoring tool were on average 25% faster than with the CATOMIR (see Figure 11). A t-test for dependent samples for iaTAR (M=3:53, SD=2.24) and CATOMIR (M=5:05, SD=2.97) turned out significantly different ($t(23)=2.84$, $p=0.00094$).

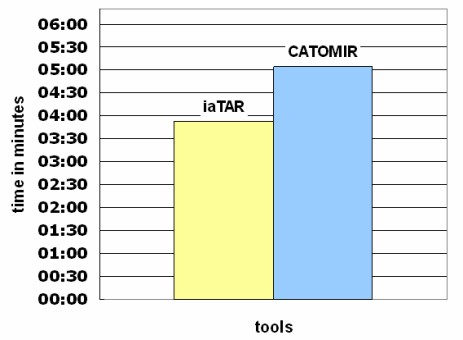

Figure 11. Duration of tasks for each tools

The number of mistakes while using the immersive authoring tool was 21, of which none required any help from the observer to correct them. In comparison, with the desktop authoring interface, the total was 36 mistakes, 21 of which couldn't be solved by the participants themselves. We consider this result shows that the immersive authoring tool is more intuitive for the users that it is easy to learn and use.

When asked about the users' preference, 42% of subjects (10 users) said they preferred using the immersive interface, while 33 % (8 users) said they would appreciate a mixture of both types, which allows the user to swap between different modes (traditional mouse-keyboard input and new Tangible AR interface). Only 25 % (6 users) said they would want to keep the mouse-keyboard interaction.

Although the test showed the efficiency and easiness of using immersive authoring interfaces for overall layout tasks, in the user interview, precise controls requiring numeric inputs still appeared to be more convenient with the 2D desktop user interfaces. However, the Tangible AR interfaces were much preferred for the tasks that include 3D spatial understandings, such as 3D rotations.

Convinced with the pilot user study, we are planning to conduct more specific user studies to investigate more detailed features of the immersive authoring approach.

6.3. Discussion

Through the cases of example application development and the user study, the proposed immersive authoring system appeared to be efficient and easy to use, yet feasible enough to create various Tangible AR applications. The concurrent testing with implementation throughout the development process appeared to be helping the developers on reducing the time for switching between implementation and testing phase.

However, currently provided logic boxes by the authoring system were not comprehensive enough for building applications with complex behaviors. In order to build applications with more complicated behaviors (or interactions), various logic box components would be necessary. A library of various logical entities is expected to be added to the authoring tool. In addition, we are also investigating interaction techniques in which users build their own custom logic boxes within the authoring environment and add them to the library for the later use.

7. Conclusion and future works

In this paper, we suggested an immersive authoring method for Tangible AR applications and described our implementation of a prototype authoring system to show its feasibility. The system used a component based application model and interaction methods, designed through analyzing the application domain. A number of development cases were described to show that our authoring system provides an efficient and easy way for constructing Tangible AR applications.

Although the data flow model between components covered the basic functions of Tangible AR applications, we also plan to investigate other behavior models such as event driven models for future support. We are also investigating inclusion of motion capture functions, so that the users could describe complicated animations by demonstration.

Other interaction methods for authoring tasks are also in need of testing. For instance, we are expecting natural gestures for controlling and authoring virtual object behaviors. Using keyboards and other

conventional user interfaces within AR systems for programming tasks is another interesting topic.

With additional research on application models and interactions, the authors are convinced that immersive authoring has a bright future as a development method for augmented reality applications.

8. References

[1] ARToolKit. http://www.hitl.washington.edu/artoolkit

[2] M. Billinghurst, I. Poupyrev, H. Kato and R. May, "Mixing Realities in Shared Space: An Augmented Reality Interface for Collaborative Computing", *Proceedings of IEEE International Conference on Multimedia and Expo (ICME 2000)*, New York, U.S.A., Jul.30-Aug.2, 2000, pp. 1641-1644.

[3] M. Billinghurst, H. Kato and I. Poupyrev, "The MagicBook – Moving Seamlessly between Reality and Virtuality", *IEEE Computer Graphics and Applications, 21*(3), May, 2001, pp. 6-8.

[4] J. Butterworth, A. Davidson, S. Hench and T. M. Olano, "3DM: A Three Dimensional Modeler Using a Head-Mounted Display", *Proceedings of Symposium on Interactive 3D Graphics*, Cambridge, Massachusetts, U.S.A., 1992, pp. 135-138.

[5] S. Güven and S. Feiner, "Authoring 3D Hypermedia for Wearable Augmented and Virtual Reality", *Proceedings of IEEE International Symposium on Wearable Computers (ISWC'03)*, New York, U.S.A., Oct.21-23, 2003, pp. 118-126.

[6] M. Haringer and H. T. Regenbrecht, "A Pragmatic Approach to Augmented Reality Authoring", *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR'02)*, Darmstadt, Germany, Sept.30-Oct.1, 2002, pp. 237-245.

[7] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto and K. Tachibana, "Virtual Object Manipulation on a Table-Top AR Environment", *Proceedings of the International Symposium on Augmented Reality (ISAR 2000)*, Munich, Germany, Oct.5-6, 2000, pp. 111-119.

[8] H. Kato, M. Billinghurst, I. Poupyrev, N. Tetsutani and K. Tachibana, "Tangible Augmented Reality for Human Computer Interaction", *Proceedings of Nicograph 2001*, Nagoya, Japan, 2001.

[9] H. Ishii and B. Ullmer, "Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms", *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, Atlanta, Georgia, U.S.A., Mar.22-27, 1997, pp. 234-241.

[10] R. S. Kalawsky, *The Science of Virtual Reality and Virtual Environments*, Addison-Wesley, 1993, pp. 212-219.

[11] G. A. Lee, G. J. Kim and C. M. Park, "Modeling Virtual Object Behavior within Virtual Environment", *Proceedings of ACM Symposium on Virtual Reality Software and Technology (VRST 2002)*, Hong Kong, China, Nov. 11-13, 2002, pp. 41-48.

[12] G. A. Lee, M. Billinghurst, G. J. Kim, "Occlusion based Interaction Methods for Tangible Augmented Reality Environments", *Proceedings of ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry (VRCAI 2004)*, NTU, Singapore, Jun.16-18, 2004, pp. 419-426.

[13] J. Liang and M. Green, "JDCAD: A Highly Interactive 3D Modeling System", *Computer & Graphics, 18(4)*, 1994, pp. 499-506.

[14] B. MacIntyre, M. Gandy, J. Bolter, S. Dow and B. Hannigan, "DART: The Designer's Augmented Reality Toolkit", *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR'03)*, Tokyo, Japan, Oct.7-10, 2003, pp.329-330.

[15] M. R. Mine, "ISSAC: A Meta-CAD System for Virtual Environments", *Computer-Aided Design, 29(8)*, August, 1997, pp. 547-553.

[16] W. Piekarski and Bruce H. Thomas, "Interactive Augmented Reality Techniques for Construction at a Distance of 3D Geometry", *Proceedings of Immersive Projection Technology / Eurographics Virtual Environments Conference (IPT/EGVE 2003)*, Zurich, Switzerland, May 22-23, 2003.

[17] I. Poupyrev, D. S. Tan, M. Billinghurst, H. Kato, H. Regenbrecht and N. Tetsutani, "Developing a Generic Augmented Reality Interface", *IEEE Computer, 35*(3), March, 2002, pp. 44-50.

[18] A. Steed and M. Slater, "Dataflow Representation for Defining Behaviours within Virtual Environments", *Proceedings of Virtual Reality Annual International Symposium (VRAIS'96)*, Mar.30–Apr.3, 1996, pp. 163-167.

[19] R. Stiles and M. Pontecorvo, "Lingua Graphica: A Visual Language for Virtual Environments", *Proceedings of IEEE Workshop on Visual Languages*, Sept.15-18, 1992, pp. 225-227.

[20] G. Wesche and H. Seidel, "FreeDrawer – A Free-Form Sketching System on the Responsive Workbench", *Proceedings of ACM Symposium on Virtual Reality Software and Technology (VRST 2001)*, Alberta, Canada, Nov.15-17, 2001, pp. 167-174.

[21] J. Zauner, M. Haller, "Authoring of Mixed Reality Applications including Multi-Marker Calibration for Mobile Devices", *Proceedings of 10th Eurographics Symposium on Virtual Environments (EGVE 2004)*, Grenoble, France, Jun.8-9, 2004, pp. 87-90.